LISTING OF THE CLAIMS

A detailed listing of claims is presented below. Please amend currently amended claims as indicated below including substituting clean versions for pending claims with the same number. In addition, clean text versions of pending claims not being currently amended that are under examination are also presented. It is understood that any claim presented in a clean version below has not been changed relative to the immediate prior version.

1. (Currently Amended) A method of computation comprising:

a) performing a critical task using an operation that is speculative while a condition of a processor used for performing said critical task is unknown;

b) in parallel with a), determining said condition;

c) if said condition is as expected, committing a first result from said performing said critical task; and

d) if said condition is not as expected, allowing said first result to benignly fail, changing said condition to be as expected by setting preconditions in virtual memory, re-performing said critical task using said operation, and committing a second result from said re-performing said critical task.

2. (Original) The method of computation as described in Claim 1, wherein said operation is a load.

3. (Original) The method of computation as described in Claim 1, wherein a) further comprises:

assigning a speculative attribute to said operation.

4. (Currently Amended) The method of computation as described in Claim 1, wherein said condition is an enablement state of said virtual memory.

5. (Original) The method of computation as described in Claim 4, wherein b) comprises:

determining, in parallel to a), if data translation is enabled, wherein when data translation is enabled said condition is as expected, and when data translation is disabled said condition is not as expected.

6. (Canceled) Please cancel Claim 6 without prejudice.

7. (Original) The method of computation as described in Claim 1, wherein a) further comprises:

receiving an interruption at an interruption handler to perform said critical task.

8. (Original) The method of computation as described in Claim 7, wherein said interruption handler is a first-

level interruption handler operating in a lightweight interruption environment.

9. (Original) The method of computation as described in Claim 1, wherein c) further comprises:

storing said first result in a virtual memory address; and wherein d) further comprises:

storing said second result in said virtual memory address.

10. (Original) The method of computation as described in Claim 1, wherein d) further comprises:

assigning a non-speculative attribute to said operation, duplicating code used to execute a) forming a duplicated code, and re-performing said critical task using said duplicated code.

11. (Currently Amended) A method of computation comprising:

a) receiving an interruption from an application to perform a critical task; and

b) incorporating speculative features of a processor to perform said critical task, that references virtual memory addresses that are known and valid, but a condition of a processor used for performing said critical task is unknown; and

c) using a speculative load to perform said critical task, and wherein c) further comprises:

c1) performing said critical task;

c2) in parallel with c1) determining said condition;

c3) if said condition is as expected by determining virtual memory is enabled, committing a first result from said performing said critical task; and

c4) if said condition is not as expected by determining virtual memory is disabled, changing said condition to be as expected, and re-performing said critical task successfully while allowing said first result to benignly fail.

12. (Canceled) Please cancel Claim 12 without prejudice.

13. (Currently Amended) The method of computation as described in Claim [[12]] 11, wherein c4) further comprises:

committing a second result from said re-performing said critical task.

14. (Original) The method of computation as described in Claim 11, wherein said interruption is handled by an interrupt handler.

15. (Canceled) Please cancel Claim 15 without prejudice.

16. (Original) A method of computation comprising:

a) receiving an interruption from an application to perform a critical task;

b) performing said critical task using a load that is speculative while a condition of virtual memory is unknown;

c) in parallel with b), determining said condition;

d) if virtual memory is enabled, committing a first result from said performing said critical task;

e) if virtual memory is disabled, enabling said virtual memory in order to re-perform said critical task successfully while allowing said first result to benignly fail since said load is speculative.

17. (Original) The method of computation as described in Claim 16, wherein e) further comprises:

e2) re-performing said critical task using said load; and

e3) committing a second result from said re-performing said critical task.

18. (Original) The method of computation as described in Claim 16, wherein a) further comprises:

assigning a speculative attribute to said load.

19. (Original) The method of computation as described in Claim 16, wherein an interruption handler receives said interruption and performs said critical task.

20. (Original) The method of computation as described in Claim 16, wherein said interruption handler operates in a lightweight interruption environment.

21. (Original) The method of computation as described in Claim 16, wherein d) further comprises:

d1) assigning a non-speculative attribute to said load, forming a non-speculative load;

d2) duplicating code used to execute b), forming a duplicated code; and

d3) re-performing said critical task with said duplicated code using said non-speculative load.

22. (Currently Amended) A computer system comprising:

a processor; and

a computer readable memory coupled to said processor and containing program instructions that, when executed, implement a method of computation comprising:

a) performing a critical task using an operation that is speculative while a condition of a processor used for performing said critical task is unknown;

b) in parallel with a), determining said condition;

c) if said condition is as expected, committing a first result from said performing said critical task; and

d) if said condition is not as expected, allowing said first result to benignly fail, changing said condition to be as expected by setting preconditions in virtual memory, re-performing said critical task using said operation, and committing a second result from said re-performing said critical task.

23. (Original) The computer system as described in Claim 22, wherein said operation is a load.

24. (Original) The computer system as described in Claim 22, wherein a) in said method further comprises:

assigning a speculative attribute to said operation.

25. (Currently Amended) The computer system as described in Claim 22, wherein said condition is an enablement state of said virtual memory.

26. (Original) The computer system as described in Claim 25, wherein b) in said method comprises:

determining, in parallel to a), if data translation is enabled, wherein when data translation is enabled said condition is as expected, and when data translation is disabled said condition is not as expected.

27. (Canceled) Please cancel Claim 27 without prejudice.

28. (Original) The computer system as described in Claim 22, wherein a) in said method further comprises:

receiving an interruption at an interruption handler to perform said critical task.

29. (Original) The computer system as described in Claim 28, wherein said interruption handler is a first-level interruption handler operating in a lightweight interruption environment.

30. (Original) The computer system as described in Claim 22, wherein c) in said method further comprises:

storing said first result in a virtual memory address; and wherein d) further comprises:

storing said second result in said virtual memory address.

31. (Original) The computer system as described in Claim 22, wherein d) in said method further comprises:

assigning a non-speculative attribute to said operation, duplicating code used to execute a) forming a duplicated code, and re-performing said critical task using said duplicated code.